L Number	Hits	Search Text	DB	Time stamp
-	553	trap$1 with key$1	USPAT; US-PGPUB	2004/08/15 13:14
-	430	(trap$1 with key$1) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 14:26
-	21	((trap$1 with key$1) and @ad<20010510) and (message$1 adj2 key$1)	USPAT; US-PGPUB	2004/08/15 13:14
-	19	(((trap$1 with key$1) and @ad<20010510) and (message$1 adj2 key$1)) and relation$4	USPAT; US-PGPUB	2004/08/15 13:18
-	270	((trap$1 with key$1) and @ad<20010510) and compar$4	USPAT; US-PGPUB	2004/08/15 13:18
-	18	(((trap$1 with key$1) and @ad<20010510) and compar$4) and (compar$4 near5 trap$1)	USPAT; US-PGPUB	2004/08/15 13:33
-	227	(messag$3 near6 event$1) with compar$3	USPAT; US-PGPUB	2004/08/15 13:33
-	145	((messag$3 near6 event$1) with compar$3) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 13:34
-	7	(((messag$3 near6 event$1) with compar$3) and @ad<20010510) and trap$1	USPAT; US-PGPUB	2004/08/15 13:34
-	3	("5355327" | "5941996" | "5966714").PN.	USPAT	2004/08/15 13:38
-	2185	compar$3 near8 trap$	USPAT; US-PGPUB	2004/08/15 14:26
-	1657	(compar$3 near8 trap$) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 14:34
-	404	((compar$3 near8 trap$) and @ad<20010510) and event$1	USPAT; US-PGPUB	2004/08/15 14:26
-	343	(((compar$3 near8 trap$) and @ad<20010510) and event$1) and (key$1 or ID$1 or number$1)	USPAT; US-PGPUB	2004/08/15 14:49
-	103	((((compar$3 near8 trap$) and @ad<20010510) and event$1) and (key$1 or ID$1 or number$1)) and agent$1	USPAT; US-PGPUB	2004/08/15 14:33
-	43	(delet$3 or discard$3) with (same$1 adj2 event$1)	USPAT; US-PGPUB	2004/08/15 14:50
-	21	((delet$3 or discard$3) with (same$1 adj2 event$1)) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 14:49
-	3115	same$1 near6 trap$1	USPAT; US-PGPUB	2004/08/15 14:49
-	2527	(same$1 near6 trap$1) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 14:54
-	1752	((same$1 near6 trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)	USPAT; US-PGPUB	2004/08/15 14:55
-	279	(((same$1 near6 trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)) and (delet$3 or discard$3)	USPAT; US-PGPUB	2004/08/15 14:50
-	77	((((same$1 near6 trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)) and (delet$3 or discard$3)) and agent$1	USPAT; US-PGPUB	2004/08/15 14:54
-	0	(((((same$1 near6 trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)) and (delet$3 or discard$3)) and agent$1) and (delet$3 with (same$1 adj6 trap$1))	USPAT; US-PGPUB	2004/08/15 14:52
-	9443	filter$3 with trap$1	USPAT; US-PGPUB	2004/08/15 14:54
-	7239	(filter$3 with trap$1) and @ad<20010510	USPAT; US-PGPUB	2004/08/15 14:54
-	4981	((filter$3 with trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)	USPAT; US-PGPUB	2004/08/15 14:55
-	105	(((filter$3 with trap$1) and @ad<20010510) and (key$1 or ID$1 or number$1)) and (compar$3 near6 trap$1)	USPAT; US-PGPUB	2004/08/15 14:55

US-PAT-NO: 6182157

DOCUMENT-IDENTIFIER: US 6182157 B1

TITLE: Flexible SNMP trap mechanism

---------- KWIC ---------

Application Filing Date - AD (1):
19960919

Brief Summary Text - BSTX (3):
Network management systems are employed to monitor, interpret, and control the operations of a network. In a typical network management system, network devices (e.g., servers, gateways, hosts) are provided with **agent** software (an "**agent**") that monitors and accumulates operational data and detects exceptional events. A management station includes management software (a "manager") at the application level which requests operational data or receives event notifications from the **agent** using management protocols. The management station is further equipped to interpret the operational data and event information to effect control of the network operations.

Brief Summary Text - BSTX (4):
Simple Network Management Protocol (SNMP) (J. Case et al., "A Simple Network Management Protocol", RFC 1157, May 1990) defines a standard protocol for the communication of management information. SNMP specifies the format and meaning of messages exchanged between managers and **agents** and the representation of names and values in those messages. A virtual information store, termed a Management Information Base (MIB) (K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, May 1990), defines the management data or objects by specifying the data variables a network device must keep, including the names of the data variables and the syntax used to express those names. For example, the MIB may specify data variables which track statistics on the status of network interfaces, incoming and outgoing datagrams, and the **number** of routing failures. The rules used to define and identify MIB data variables are provided by the Structure of Management Information (SMI) specification (M. Rose and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, May 1990). Each managed data variable or object has a name known as an object identifier which specifies an object type. The object type together with an object instance uniquely identifies a specific instantiation of the object. For convenience, a text string known as an object descriptor is used to refer to the object type.

Brief Summary Text - BSTX (5):
SNMP uses a fetch-store paradigm to effect operations between a manager and

agent. Specifically, SNMP defines get-request, get-next-request, get-response, and set-request commands which provide the basic fetch and store operations. In addition, SNMP defines a trap command by which an **agent** asynchronously sends information to a manager triggered by an event. Thus, a management station requests operational data or receives event notifications by means of this simple set of SNMP commands.

Brief Summary Text - BSTX (6):

A limitation of known MIB structures is that trap definitions are predefined in the MIB. For any particular MIB, traps are defined which trigger when specific conditions are met. Since the traps are predefined at the time the MIB is designed, a network management station typically must poll the network device for values of MIB variables not specified in a defined trap. Polling across the network is undesirable since it adds to network traffic. To provide for more extensive monitoring of events without having to poll, a MIB could be designed to include additional predefined trap definitions, one for each combination of variables. However, for large MIBs, the **number** of variables needed can be prohibitive for this approach. Further, since the traps are predefined, a user of the management station does not have the option of turning certain traps on and off.

Brief Summary Text - BSTX (8):

However, the RMON MIB is limited in the variables that it supports and the conditions that it can specify for generating a trap. While the RMON MIB supports thresholding on statistical values (e.g., the **number** of packets or collisions counted for a monitored interface), it has limitations on trapping multistate or enumerated status variables. For example, a status variable for a device typically may have multiple states defined, such as "unknown", "running", "warning", "testing", and "down". Each state is represented in the MIB object syntax by a different integer value, e.g., unknown=1, running=2, warning=3, testing=4, and down=5. Since the RMON MIB only supports thresholding using "greater than or equals to" and "less than or equals to" comparisons, thresholding for a particular state of a multistate status variable is difficult and cumbersome.

Brief Summary Text - BSTX (15):

According to another aspect of the invention, a trap defined by an alarm threshold record can persist over a restart of the network device or **agent**.

Brief Summary Text - BSTX (16):

According to still another aspect of the invention, the threshold record includes a descriptor variable comprising a descriptor string for authenticating a trap over restart of the network device or **agent**.

Drawing Description Text - DRTX (4):

FIG. 2 is a block diagram showing the relationship among the **agent**/MIBs in a preferred embodiment of the invention.

Drawing Description Text - DRTX (6):
FIG. 4 is a flow diagram of the comparison execution thread of the svrmgt **agent** in accordance with the present invention.

Drawing Description Text - DRTX (7):
FIG. 5 is a flow diagram of the threshold execution thread of the svrmgt **agent** in accordance with the present invention.

Detailed Description Text - DETX (3):
The management station 10 includes a management processor 12 and a terminal 14. The terminal 14 provides a user interface for a user to interact with a management software application 16 running on management processor 12. Each of the network elements includes an **agent** software application 26, 28, and 30, respectively, which monitors and accumulates operational data and detects exceptional events. The manager 16 requests the operational data or receives event notifications from the **agents** 26, 28, 30 using the SNMP management protocol across the network connections 32, 34, 36. The management station is further equipped to interpret the operational data and event information to effect control of network operations, which is an area that is beyond the scope of this specification.

Detailed Description Text - DETX (4):
In the preferred embodiment of the present invention, the server 22 and management station 10 run a Microsoft Windows NT platform version of ServerWORKS Manager 2.0 software provided by Digital Equipment Corporation, assignee of the present invention. The invention can also be implemented on other platforms including SCO Unix, Novell Netware, and others. It should be noted that the SNMP **agents,** described further below, are independent of the management software running on the management station 10.

Detailed Description Text - DETX (5):
As noted above, an **agent** uses a MIB which defines management data by specifying the data variables a network device must keep, including the names of the data variables and the syntax used to express those names. The present invention is concerned with providing a flexible and extensible mechanism for monitoring variables and conditions associated with a network element such as server 22. In order to provide such a mechanism, the preferred embodiment defines a so-called "private" or extension MIB called the svrmgt MIB 41.

Detailed Description Text - DETX (6):
The svrmgt MIB 41 is accessible by the **agent** 28 running on the server 22. FIG. 2 shows the **agent** 28 which comprises several **agents** working together. In the preferred embodiment of the invention, the **agent** 28 includes a Windows NT SNMP **agent** 29 provided by Microsoft Corporation. The NT SNMP **agent** 29 is a so-called extensible **agent** which transmits and receives SNMP messages over the network to and from the manager 16 running on the management processor 12. **Agents,** designated as svrmgt, host resources, svrsys, and ntcmgt **agents** then

function as extension **agents** of the NT SNMP 29 **agent** by receiving requests, completing the requests, and sending information back to the NT SNMP **agent** 29. The svrmgt **agent**/MIB 40 provides thresholding of variables associated with a host resources **agent**/MIB 42, a svrsys **agent**/MIB 44, and a ntcmgt **agent**/MIB 46. The host resources MIB 43 is a predefined standard MIB (P. Grillo and S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993) that defines data variables for managing host computers. Managed data variables that are defined by the host resources MIB 43 include the length of time the system has been up; the **number** of errors on a particular device, such as a hard drive; the load on a particular processor; printer status; and the type and status of installed software. The svrsys and ntcmgt MIBs 45, 47 are "private" or extension MIBs defined by Digital Equipment Corporation, assignee of the present invention. The svrsys MIB 45 defines hardware specific information for both Intel and Alpha processor platforms. Managed data variables include the type of system, such as Windows NT; the type of processor and processor utilization; memory configuration, type, and available paging memory; thermal and voltage sensor information; and fan status. The ntcmgt MIB 47 supports NT cluster software and defines data variables such as software status; group control information; type of policy for selecting a primary server; and the object type (e.g., disk, sql) for a particular object.

Detailed Description Text - DETX (9):

In the svrAlarms group 52, a svrThresholdTable 56 provides a table of thresholds against which the svrmgt **agent** 40 monitors. The svrThresholdTable table 56 describes conditions (i.e. thresholds) for setting and resetting alarms. The thresholds are defined in records, one threshold record per svrThresholdEntry 58 of the svrThresholdTable 56. A svrAlarmNextThrIndex variable 55 is used to name the instances or records of the svrThresholdTable 56. The svrThresholdEntry 58 is a record which contains the following variables as illustrated in FIG. 3B (A program listing of the svrThresholdTable 56 is provided further below):

Detailed Description Text - DETX (16):

svrThrPersistent 72: a flag which when set (i.e. has value "true"), the threshold persists across restart of the **agent**.

Detailed Description Text - DETX (29):

As noted above, each threshold defined using the svrmgt MIB 41 structure constitutes a record or instance. An array of pointers is created for pointing to the threshold records in memory. The svrmgt **agent** 40 includes three different threads of execution: main, comparison, and threshold threads. The main thread performs the SNMP requests on the records in the threshold array. The comparison thread, described further below, is initiated on **agent** restart and implements the persistence aspect of the present invention. The threshold thread, also described further below, works on records in the threshold array that have been enabled, performing polling and comparison functions and sending traps as needed.

Detailed Description Text - DETX (30):

The comparison and threshold execution threads of the svrmgt **agent** 40 can be better understood by illustrating an example threshold record for sending a trap in the case of a failed CPU board fan on the server 22. In that case, fan status equals "failed". Referring again to FIG. 1, a user at management station 10 enters commands in a higher level application program which the manager 16 translates to SNMP get and set commands towards the server 22. In the first step, the value of the svrAlarmNextThrIndex 55 provides a **key** which is used to identify the subject threshold record 58 in the svrThresholdTable 56. Using this **key** to identify the threshold table entry, the following threshold record 58 attributes can be set:

Detailed Description Text - DETX (43):

An important aspect of the present invention is the persistence feature which provides for verification of instances or records in the svrThresholdTable 56. In the preferred embodiment, the persistence feature is implemented using the svrThrPersistent, svrThrComparisonName, and svrThrComparisonValue variables. When the threshold record 58 is created, the management station 10 gets the value of the object identifier pointed to by the svrThrComparisonName variable 92. The svrThrComparisonName variable 92 points to an ASCII text string that can be used to uniquely identify the item or instance being thresholded. That ASCII text string value is then placed into the svrThrComparisonValue variable 94 using a set command. Referring now to FIG. 4, a flow diagram of the comparison execution thread of the svrmgt **agent** 40 is shown and will be described in relation to the above example record 58. Upon a restart of the **agent** on server 22, the svrmgt **agent** 40 steps through threshold records 58. For each record, the **agent** at step 100 gets the current value of the object identifier pointed to by the svrThrComparisonName variable 92. The **agent** then compares the current value of svrThrComparisonName 92 with the svrThrComparisonValue 94 at step 102 to ensure that the correct object is being polled. For example, consider the case where the fan status threshold had been set for a second CPU board, the server had been shut down, the board removed, and the server rebooted. In that case, the value comparison of the current value of svrThrComparisonName 92 with the svrThrComparisonValue 94 would be false and the svrmgt **agent** 40 would mark the record 58 to an "in error" state at step 104.

Detailed Description Text - DETX (44):

Referring to FIG. 5, a flow diagram of the threshold execution thread of the svrmgt **agent** 40 is shown and will now be described in relation to the above example record 58.

Detailed Description Text - DETX (45):

The svrmgt **agent** 40 begins polling by getting the current value for the attribute pointed to by svrThrVariableName variable 64 at step 200. The svrThrLastValue variable 78 is then set to the current sample value at step 202. At step 204, the svrmgt **agent** 40 compares that value to the value of the svrThrThresholdValue variable 74 using the mathematical operator that was set in the svrThrAlarmType variable 68. In the example, the mathematical operator is an equalTo operator, which means that if the fan status equals failed (svrThresholdvalue=4), a possible alarm condition exists. Before a trap can be

sent, however, the alarm state is checked at step 210. If the state of svrThrAlarmState 80 is in the reset state, then at step 212 the alarm state is changed to the set state. The svrmgt **agent** 40 then checks whether traps are enabled at step 214. If traps are enabled, a trap is sent at step 216 at the severity level set in the record 58 by the svrThrSeverity variable 96. In the example, a high level trap is sent to the management station. Thus, the svrThrSeverity variable 96 enables different severity levels to be indicated in different traps. Upon receipt of the trap, the management station 10 interprets the severity level of the trap and responds appropriately. In contrast, traps of the prior art lacked any indication of degree or level of severity.

Detailed Description Text - DETX (47):
In the example, if the fan status no longer equals failed(4), then svrThrAlarmState 80 is set to the reset state. Polling continues and if the fan status becomes failed again, another trap would be sent following the steps 210, 212, 214, and 216. This hysteresis mechanism provides an accurate or "true second" **trap transmission as opposed to a repeated trap message on the same** device state polled multiple times before the condition is remedied. It is important to note that such hysteresis on a status variable is not known to be possible with the RMON MIB.

Detailed Description Text - DETX (48):
There are applications where it can be advantageous to not implement hysteresis. For example, after an event such as a device failure has occurred and an initial trap has been sent, corrective action may be initiated at the management station. During the corrective action, additional **trap messages associated with the same** triggering event would be useful feedback to indicate whether and when the corrective action is successful. One way to avoid the hysteresis is for the svrmgt **agent** 40 to ignore the svrThrAlarmState alarm state variable 80.

Detailed Description Text - DETX (50):
The preferred embodiment has been shown with reference to a svrmgt MIB 41 that supports monitoring of three specific MIBs. It will be apparent to one skilled in the art that the principles of the invention can be readily applied to embodiments which support monitoring of any **number** and variety of MIBs.

Detailed Description Text - DETX (57):
"Table of thresholds against which the **agent** should check for exceptions. This table describes conditions for setting and resetting alarms. Alarms may be set on absolute values--i.e. the current integer value of the sampled variable--or on delta values--i.e. the difference between the current or last value. Alarms may be GreaterThan exception alarms, LessThan exception alarms, EqualsTo alarms etc. See svrThrAlarmType for differences. Hysteresis is introduced by providing thresholds both for setting and resetting of the alarm state, thereby limiting the **number** of traps that will be sent on alarm triggering. Alarms may be created to persist across **agent** reboots, but this is not recommended for dynamic table variables. The triggering of an alarm will

change a state variable in the conceptual row. It may also trigger the sending of a trap, the local logging of an event, or the triggering of a locally-defined action."

Detailed Description Text - DETX (132):
To **delete** the row, set the status to rowinvalid. It is a local implementation matter whether the row is actually removed from the table. Management applications must be prepared to ignore rows with a status of rowinvalid.

Detailed Description Text - DETX (133):
Errors in variable polling and threshold checking that are determined by local implementation to be non-correctable will cause a row status change to rowError. Once the status is set to rowError by the **agent, the agent** will not reset it. Instead, it is up to the management console to reset the status based on information returned via svrThrErrorValue or for other reasons."

Detailed Description Text - DETX (140):
"The OID of an integer-valued variable to be tested against this threshold. On row creation, this variable will equal the value 0.0, and must be set to the OID of an integer-valued variable before enabling the alarm. It's possible to get an error setting this due to invalid oid (may not support thresholding on this oid due to data type perhaps) or that we couldn't access the **agent** that supports this oid."

Detailed Description Text - DETX (184):
"True if this threshold should persist across **agent** restarts. Default value on row creation is FALSE."

Detailed Description Text - DETX (214):
"Whether this alarm is currently set or reset. This variable is used by polling management applications to determine if a threshold exception state has been detected based on this alarm definition. It will initially have a value of reset when the alarm is enabled or the **agent** is restarted. For state change rules look at the definition for svrThrAlarmType.

Detailed Description Text - DETX (243):
"This attribute is used to describe the type of threshold. This is set by the management console not by the **agent.**"

Detailed Description Text - DETX (258):
"An OID to a descriptor attribute which can be used with persistent to verify that the svrThrVariableName instance is correct. On **agent** restarts the value for this oid will be retrieved and compared to the svrThrComparisonValue. If not equal, then it's possible that the oid instancing for svrThrVariableName is incorrect.

Detailed Description Text - DETX (266):
"Data value of svrThrComparisonName--used when persistent is set. This value is compared to the current value on **agent** restarts. This attribute is optional. The default value is NULL"

Claims Text - CLTX (2):
coupling an **agent** to the selected network device for monitoring current status of a device object and communicating the status to the system management station;

Claims Text - CLTX (3):
providing a data memory member accessible by the **agent** for storing attributes of the device object in a threshold record;

Claims Text - CLTX (6):
upon a restart of the **agent,** verifying persistence of the threshold record by:

Claims Text - CLTX (13):
coupling an **agent** to the network device for monitoring current status of a device object and communicating the status to the system management station, wherein the device object comprises a multistate status variable having at least two enumerated states, each state represented by an integer data value;

Claims Text - CLTX (14):
providing a data memory member accessible by the **agent** having a user-definable threshold value associated with the device object;

US-PAT-NO: 6772093

DOCUMENT-IDENTIFIER: US 6772093 B2

TITLE: Equipment inspection and evaluation system, equipment management system, and computer-readable record medium with equipment management program stored therein

---------- KWIC ---------

Application Filing Date - AD (1):
20010124

Brief Summary Text - BSTX (46):
According to this feature, a representation, e.g. a diagram, of the equipment is displayed on the display screen, and symbols, e.g. icons, are disposed on the equipment diagram to indicate that devices corresponding to the respective icons are disposed in the equipment at locations corresponding to the locations displayed on the diagram on the screen. Desired ones of the icons are selected through the symbol selecting section. The second display control section calls detailed data of the devices corresponding to the selected icons from the detailed data memory section and displays the called detailed data on the display screen. Thus, **relationship** in position among the respective devices in the equipment and detailed data of the devices can be readily grasped on the display screen.

Drawing Description Text - DRTX (8):
FIG. 7 shows how to operate **keys** on a keyboard of the inspection and evaluation system in order to set **trap** data of a desired **trap** in the preset region, and also a form of display given in a display section of the inspection and evaluation system shown in FIG. 5.

Drawing Description Text - DRTX (10):
FIG. 9 shows how to operate **keys** on a keyboard of the inspection and evaluation system in order to call desired **trap** data stored in the preset region, and also a form of display given in a display section of the inspection and evaluation system shown in FIG. 5.

Detailed Description Text - DETX (15):
Writing of **trap** data into the preset region 162 and calling or reading desired **trap** data from the preset region 162 is carried out by the CPU 13 in accordance with a **key** entry through the data entry section 18. The CPU 13 also causes a **message based on the key** entry to be displayed on the display 17.

Detailed Description Text - DETX (18):

As previously stated, commands for writing and reading desired trap data in and from the preset region 162 are give through the data entry section 18. The keys on the data entry section 18 are arranged as shown in FIG. 5 which is a front view of the inspection and evaluation apparatus 12. The **keys** are sorted into a power switch **key** group 181, a function **key** group 182, a **trap** type selecting **key** group 183, and a numerical **key** group 184. The display 17 is disposed in the top portion above these key groups, and may be a liquid crystal display panel which can display a message in, for example, two rows. The inspection and evaluation apparatus 12 is generally rectangular and has such a size that it can be held by hand. The inspection and evaluation apparatus 12 has an input terminal 12a at the top end surface for connecting the apparatus 12 to the probe 11 via the cable 11a.

Detailed Description Text - DETX (19):

Next will be described, how to manipulate the **keys** on the data entry section 18 and how the CPU 13 operates for writing desired **trap** data into the preset region 162, using the **trap** codes, with reference to FIGS. 6 and 7.

Detailed Description Text - DETX (20):

FIG. 6 is a state transition diagram showing the operation of the CPU 13 when trap data is written in and read from the preset region 162. FIG. 7 illustrates the sequence of operating the **keys** on the data entry section 18 for writing **trap** data into the preset region 162, and also the messages on the display 17.

Detailed Description Text - DETX (22):

Next, the type of the **trap** is selected by pressing an appropriate **key in the trap** type selecting **key** group 183. Then, the CPU 13 enters into a model writing mode M4 and causes the display 17 to display, after the indication of "MODEL" in the lower row, a two-digit number and a **trap** type selected through the **trap** type selecting **key** 183, as shown in FIG. 7, Part (b). FIG. 7, Part (b) shows that the "FLOAT" **key in the trap** type **key** group 183 was pressed. If it is desired to change the **trap** type to another type from the FLOAT type, the **key** for the desired type is pressed.

Detailed Description Text - DETX (24):

Keys with arrows ".uparw." and ".dwnarw." indicated on their surfaces in the function key group 182 are pressed to enter one of numbers 00 through 30 corresponding to a desired smaller memory region. In other words, one of the smaller memory region 162b in which to store desired **trap** data is selected by operating the ".uparw." and ".dwnarw." **keys**. For example, the ".uparw." key may be pressed once to select a first smaller memory region numbered "01", which may be referred to as memory number. In this case, the message displayed is as shown in FIG. 7, Part (c). Below the memory number (or in the first digit of the memory number) on the display 17, a cursor 17a blinks, indicating that the digit can be changed. It should be noted that the memory number "00" does not represent a smaller memory region 162b, but it is a kind of message to indicate that the CPU 13 is now in the model writing mode M4. Therefore, no

trap data can be written in this memory number "00".

Detailed Description Text - DETX (25):
After selecting the memory number, numeral **keys** are used to enter the **trap** code of a **trap of which trap** data should be written, beginning with the digit in the highest position toward the digit in the lowest position, e.g. from the thousands digit, the hundreds digit, the tens digit down to the units digit. When the thousands digit is entered, the CPU 13 enters into a trap code entry mode M6. The message on the display 17 displayed when the thousands digit of, for example, "1" is entered is shown in FIG. 7, Part (d). It is seen that the cursor 17a, too, has moved to the position below the thousands digit.

Detailed Description Text - DETX (27):
The trap code for a particular trap can be known from a table containing trap codes shown in **relation** to corresponding trap models.

Detailed Description Text - DETX (29):
Then, when the CPU 13 is in the state shown in FIG. 7, Part (e), an "ENT" **key** in the numeral **key** group 184 is pressed, the CPU reads the **trap** data corresponding to the entered **trap,** i.e. the **trap** data for the **trap** of which the model is "J3X-2" in the illustrated example, from the **trap** data memory region 161. The read trap data is written in the first memory region 162b. Then, the message on the display 17 changes to a message indicating that the writing of the trap data has been finished. This message is shown in FIG. 7, Part (f). The CPU 13 returns to the idling mode M2.

Detailed Description Text - DETX (30):
Alternatively, the **trap** data can be written by pressing the ".uparw." and **keys** when the apparatus is in the state shown in FIG. 7, Part (e). In this case, upon pressing the arrowed key, the message on the display 17 returns to the state shown in FIG. 7, Part (c).

Detailed Description Text - DETX (31):
Further, if it is desired in the state shown in FIG. 7, Part (e) to alter the **trap** to be written, the **trap** code for the desired **trap** is entered by pressing appropriate numerical **keys,** which returns the apparatus 12 to the state shown in FIG. 7, Part (d).

Detailed Description Text - DETX (33):
In the state of FIG. 7, Part (g), if, for example, the "ENT" **key** is pressed, the writing of **trap** data into the first **trap** memory region 162b or the renewal of **trap** data in the first memory region 162b is not done.

Detailed Description Text - DETX (34):
According to the data writing method thus far described with reference to FIG. 7, a **trap** code of a desired **trap** is entered directly by pressing **keys** on

the keyboard. However, if one does not know the trap code of the desired trap, he or she must find it out from the previously described trap code list. According to the illustrated example, in addition to the trap code entry method, a trap model retrieving and entering method is also employed. In the trap model retrieving and entering method, a trap model of the desired trap is retrieved, and the trap data for the desired trap is written on the basis of the retrieved trap model. The trap model retrieving and entering method is described in detail with reference to FIGS. 6 and 8.

Detailed Description Text - DETX (41):

In the state shown in FIG. 9, Part (c), i.e. in the model reading mode 12 shown in FIG. 6, the "ENT" **key** is pressed after the smaller memory region 162b where **trap** data for the desired **trap** is stored is selected by pressing one of the arrowed **keys**. In FIG. 9, Part (c), the selected smaller memory region 162b is the first region numbered "01" where the trap data for the Model "J3X-2" trap is contained. When the "ENT" **key** is pressed, the **trap** data stored in the selected smaller memory region is called, and the model of the **trap of which the trap** data has been called is displayed on the display 17, as shown in FIG. 9, Part (d). In the case of FIG. 9, Part (d), the trap data for Model "J3X-2" trap has been called. An operator can know the trap type and model of a trap to be inspected since they are indicated on a plate attached to the trap housing.

Detailed Description Text - DETX (62):

If a bypass valve 51 is to be evaluated instead of a **trap** 41, a **key** on the data entry section 18, e.g. the "ENT" **key** in the numerical **key** group 184, may be pressed once. This makes the CPU 13 shift into the valve inspection and evaluation mode M30 from the trap inspection and evaluation mode M20 and is ready for inspection and evaluation of a trap 51. At the same time, the display 17 displays a message indicating that the CPU 13 changes its mode from the trap inspection and evaluation mode M20 to the valve inspection and evaluation mode M30.

Detailed Description Text - DETX (64):

After the judgment step 300, the CPU 13 automatically returns to the idling state 100 and becomes ready for the next valve inspection and evaluation. Accordingly, if another bypass valve 51 should be inspected and evaluated, the probe 11 is urged against the valve surface, and the same procedure is repeated. On the other hand, if an operator wants to inspect and evaluate a **trap** 41, he presses the "ENT" **key** once, so that CPU 13 shifts from the valve inspection and evaluation mode M30 to the **trap** inspection and evaluation mode M20.

Detailed Description Text - DETX (65):

As described above, in the manual switching mode, by pressing the "ENT" **key** when the CPU 13 is in the idling state 100, the inspection and evaluation mode of the CPU 13 can be switched between the **trap** inspection and evaluation mode M20 and the valve inspection and evaluation mode M30. In other words, in the manual switching mode, unless the "ENT" key is pressed when the CPU 13 is in

the idling state 100, the inspection and evaluation mode currently employed is not switched to the other. This feature is useful for successively inspecting and evaluating either of traps 41 and valves 51.

Detailed Description Text - DETX (66):
However, in order to alternately evaluate combinations of **trap** 41 and bypass valve 51, the inspection and evaluation mode is also alternately switched between the **trap** inspection and evaluation mode M20 and the valve inspection and evaluation mode M30 by pressing the "ENT" **key** a number of times, which is a very troublesome operation.

Detailed Description Text - DETX (69):
It should be noted that in the automatic switching mode of the CPU 13, too, if the "ENT" **key** is pressed when the CPU 13 is in the idling state 100, the inspection and evaluation mode can be switched between the **trap** inspection and evaluation mode M20 and the valve inspection and evaluation mode M30.

Detailed Description Text - DETX (72):
Thus, in the automatic switching mode, the CPU 13 automatically shifts to one of the trap inspection and evaluation mode M20 and the valve inspection and evaluation mode M30 after it performs the inspection and evaluation in the other mode. Therefore, when the automatic switching mode is used for alternately inspecting and evaluating pairs of a trap and a bypass valve, there is no need for manually switching the switching mode alternately. As described above, if, in the automatic switching mode, it becomes necessary to successively evaluate two **traps** 41 or two valves 51, the "ENT" **key** is pressed when the CPU 13 is in the idling state 100, which can switch the inspection and evaluation mode from one mode to the other.

Detailed Description Text - DETX (162):
As described above, according to the invention, it is easy to grasp the positional **relationship** among traps and the detailed data of the traps by simply seeing the piping diagram 35 and the icons 36-40, for example, disposed on it.

Detailed Description Text - DETX (215):
In the described example, the order of extracted traps to be inspected is changed manually, but the re-arrangement of data may be done automatically on the basis of, for example, positional **relationship** among the traps as shown in FIG. 21. For example, traps may be arranged automatically in accordance with the distance from the entrance to a particular plant.

Detailed Description Text - DETX (218):
In the described example, the steam pressure in the interior of a trap is determined indirectly by detecting the temperature of the surface of the housing of that trap. However, if the exact steam pressure in the **trap** can be known, it may be manually input through the data entry section or **keys** 18. The

use of exact steam pressures can provide more exact trap evaluation than using indirectly obtained steam pressures. Further, if high exactness is not required in evaluation, only measurements of vibrations may be used in evaluating traps or computing the amount of steam leakage.

US-PAT-NO: 6507852

DOCUMENT-IDENTIFIER: US 6507852 B1

TITLE: Location-independent service for monitoring and alerting on an event log

---------- KWIC --------

Abstract Text - ABTX (1):
A method, apparatus, and article of manufacture for monitoring and alerting on an event log. One or more alert policies is accessed, wherein each of the alert policies is comprised of one or more rules stored on a computer. An event log stored on a computer is accessed in a location-independent manner to gather one or more event messages stored therein. The **event messages** are filtered by **comparing** them to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked.

Application Filing Date - AD (1):
20000417

Brief Summary Text - BSTX (11):
To address the requirements described above, the present invention discloses a method, apparatus, and article of manufacture for monitoring and alerting on an event log. One or more alert policies is accessed, wherein each of the alert policies is comprised of one or more rules stored on a computer. An event log stored on a computer is accessed in a location-independent manner to gather one or more event messages stored therein. The **event messages** are filtered by **comparing** them to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked.

Brief Summary Text - BSTX (14):
Each rule of an alert policy also specifies one or more Alert Actions. The Alert Action specifies what is to be done when matching errors exceed the specified limit. For example, an Alert Action may comprise: sending an email to a user-defined address, sending a page with a user-defined message to a user-defined paging service, generating a **trap,** running a user-specified program, writing a message to log, and executing a script of database commands. Each Alert Action may comprise a single defined action, or may comprise a plurality of individual actions. In addition, the Alert Action may specify a period that must expire before the Alert Action is repeated for the same event. These aspects of the alert policies are defined using an alert policy editor.

Detailed Description Text - DETX (25):
Each rule in an Alert Policy 120 also specifies one or more Alert Actions

118. The Alert Action 118 specifies what is to be done when matching errors exceed the specified limit. Examples of Alert Actions 118 include sending an email to a user-defined address, sending a page with a user-defined message to a user-defined paging service, generating an SNMP (Simple Network Management Protocol) **trap,** running a user-specified program, writing a message to log, and executing a script of database commands. An Alert Action 118 may also be defined to be a plurality of individual Alert Actions 118. In addition, the Alert Action 118 may specify a period that must expire before the Alert Action is repeated for the same event.

Detailed Description Text - DETX (40):

The Alert Action DLL 116 is a library of functions comprising an application programming interface (API) that perform various Alert Actions 118, such as sending a page, sending an email message, generating an SNMP (Simple Network Management Protocol) **trap,** writing a message to log, or executing a script of database commands. The Event Monitoring and Alerting Service 108 invokes these functions to alert users of important events (as determined by the Alert Policy 120).

Detailed Description Text - DETX (91):

This concludes the description of the preferred embodiments of the present invention. In summary, the present invention describes a method, apparatus, and article of manufacture for monitoring and alerting on an event log. One or more alert policies is accessed, wherein each of the alert policies is comprised of one or more rules stored on a computer. An event log stored on a computer is accessed in a location-independent manner to gather one or more event messages stored therein. The **event messages** are filtered by **comparing** them to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked.

Claims Text - CLTX (1):

1. A method for monitoring and alerting on an event log, comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) periodically checking the alert policies to dynamically adopt updated ones of the alert policies.

Claims Text - CLTX (3):

3. A method for monitoring and alerting on an event log, comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein the alert policies contain criteria by which the event log is searched, the alerts are raised, and the

alert actions are invoked.

Claims Text - CLTX (4):
4. A method for monitoring and alerting on an event log, comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule includes one or more defined criteria selected from a group comprising: one or more Event IDs, an Event Period indicating a period within which the event messages must occur in the event log for an alert to be raised and an alert action to be invoked, an Event Count indicating a count of the event messages that must occur within the event log within the Event Period and corresponding to the Event IDs to raise an alert and trigger an alert action, a Search Phrase that specifies one or more words that must be included within the event messages, and an Alert Any flag that determines whether or not there must be at least one occurrence of each and every Event ID that is specified by the rule.

Claims Text - CLTX (9):
9. A method for monitoring and alerting on an event log, comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule specifies an alert action selected from a group comprising sending an email to a user-defined address, sending a page with a user-defined message to a user-defined paging service, generating a **trap,** running a user-specified program, writing a message to log, and executing a script of database commands.

Claims Text - CLTX (12):
12. A method for monitoring and alerting on an event log, comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) defining the alert policies using an alert policy editor.

Claims Text - CLTX (13):
13. A computer-implemented apparatus for monitoring and alerting on an event log, comprising: (a) means for accessing one or more alert policies comprised of one or more rules stored on a computer; (b) means for accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; (c) means for filtering the **event**

messages by comparing the event messages to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) means for periodically checking the alert policies to dynamically adopt updated ones of the alert policies.

Claims Text - CLTX (15):

15. A computer-implemented apparatus for monitoring and alerting on an event log, comprising: (a) means for accessing one or more alert policies comprised of one or more rules stored on a computer; (b) means for accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) means for filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein the alert policies contain criteria by which the event log is searched, the alerts are raised, and the alert actions are invoked.

Claims Text - CLTX (16):

16. A computer-implemented apparatus for monitoring and alerting on an event log, comprising: (a) means for accessing one or more alert policies comprised of one or more rules stored on a computer; (b) means for accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) means for filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule includes one or more defined criteria selected from a group comprising: one or more Event IDs, an Event Period indicating a period within which the event messages must occur in the event log for an alert to be raised and an alert action to be invoked, an Event Count indicating a count of the event messages that must occur within the event log within the Event Period and corresponding to the Event IDs to raise an alert and trigger an alert action, a Search Phrase that specifies one or more words that must be included within the event messages, and an Alert Any flag that determines whether or not there must be at least one occurrence of each and every Event ID that is specified by the rule.

Claims Text - CLTX (21):

21. A computer-implemented apparatus for monitoring and alerting on an event log, comprising: (a) means for accessing one or more alert policies comprised of one or more rules stored on a computer; (b) means for accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) means for filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule specifies an alert action selected from a group comprising sending an email to a user-defined address, sending a page with a user-defined message to a user-defined paging service, generating a **trap,** running a user-specified program, writing a message to log, and executing a script of database commands.

Claims Text - CLTX (24):
24. A computer-implemented apparatus for monitoring and alerting on an event log, comprising: (a) means for accessing one or more alert policies comprised of one or more rules stored on a computer; (b) means for accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) means for filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) means for defining the alert policies using an alert policy editor.

Claims Text - CLTX (25):
25. An article of manufacture embodying logic for monitoring and alerting on an event log, the logic comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) periodically checking the alert policies to dynamically adopt updated ones of the alert policies.

Claims Text - CLTX (27):
27. An article of manufacture embodying logic for monitoring and alerting on an event log, the logic comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein the alert policies contain criteria by which the event log is searched, the alerts are raised, and the alert actions are invoked.

Claims Text - CLTX (28):
28. An article of manufacture embodying logic for monitoring and alerting on an event log, the logic comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule includes one or more defined criteria selected from a group comprising: one or more Event IDs, an Event Period indicating a period within which the event messages must occur in the event log for an alert to be raised and an alert action to be invoked, an Event Count indicating a count of the event messages that must occur within the event log within the Event Period and corresponding to the Event IDs to raise an alert and trigger an alert action, a Search Phrase that specifies one or more words that must be included within the event messages, and an Alert Any flag that determines whether or not there must be at least one occurrence of each and every Event ID that is specified by the

rule.

Claims Text - CLTX (33):
33. An article of manufacture embodying logic for monitoring and alerting on an event log, the logic comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and (c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and wherein each rule specifies an alert action selected from a group comprising sending an email to a user-defined address, sending a page with a user-defined message to a user-defined paging service, generating a **trap,** running a user-specified program, writing a message to log, and executing a script of database commands.

Claims Text - CLTX (36):
36. An article of manufacture embodying logic for monitoring and alerting on an event log, the logic comprising: (a) accessing one or more alert policies comprised of one or more rules stored on a computer; (b) accessing an event log stored on a computer in a location-independent manner to gather one or more event messages stored therein; and c) filtering the **event messages by comparing the event messages** to the rules of the alert policies to raise an alert and determine whether an alert action should be invoked; and (d) logic for defining the alert policies using an alert policy editor.